**Fraction Media Processing System**

**Overview**

The **Fraction Media Processing System** is a scalable and efficient media processing solution designed to handle image and video processing tasks. It leverages **Redis** for queue management, **GraphicsMagick** for image manipulation, **FFMPEG** for video processing, **Nginx** as a reverse proxy, and **PM2** for process management and monitoring.

**Features**

**Image Processing**

* Resize images to **200x200 pixels**.
* Convert images from multiple formats (JPEG, PNG, GIF, WebP) to **JPG**.
* Store processed images with **unique identifiers**.

**Video Processing**

* Extract **thumbnails** from the **5-second mark** of videos.
* Convert videos from **MP4 to AVI** format.
* Store processed videos and their corresponding thumbnails.

**Task Queue & Caching**

* **Redis-backed queuing system** to manage media processing tasks.
* **Job status tracking** (Pending, In Progress, Completed).
* Optimized resource utilization with **worker processes**.

**Scalable Architecture**

* **PM2** for process management and scalability.
* **Nginx** for efficient static file serving and reverse proxying.
* Supports **horizontal scaling** with multiple worker processes.

**System Architecture**

1. **Client Uploads Media Files** – Users upload images and videos via the API.
2. **Express.js Handles Uploads** – The backend API temporarily stores uploaded files.
3. **Redis Queue Manages Tasks** – Tasks are added to a Redis queue using Bull.
4. **Worker Processes Handle Processing** – Image and video processing happens asynchronously in worker scripts.
5. **Nginx Serves Processed Files** – Processed media files are served efficiently.
6. **PM2 Ensures Availability** – PM2 manages system processes for stability and scalability.

**System Architecture and Design Choices**

1. **Reverse Proxy and Media Serving (Nginx)**
   * Nginx acts as a reverse proxy, forwarding API requests to the backend.
   * It serves processed media files (images, videos, thumbnails) efficiently.
2. **Backend Development (Node.js & Express.js)**
   * Express.js is used to handle API requests and file uploads.
   * The backend consists of a main application and worker processes for image and video processing.
3. **Worker Processes (Scalability & Load Balancing)**
   * Image processing is handled by the imageWorker, which can be scaled horizontally.
   * Video processing follows the same model with the videoWorker.
   * Redis queue ensures proper task distribution and load balancing.

**Prerequisites**

Before installing and running the system, ensure you have the following installed:

* **Node.js** (v14+)
* **Redis Server**
* **GraphicsMagick**
* **FFMPEG**
* **Nginx**
* **Docker**
* **PM2** (globally installed)

**Installation**

**Step 1: Clone the Repository**

git clone < https://github.com/SirDylan98/FractionMediaProcessingApp.git>

cd Fraction

**Step 2: Install Dependencies**

npm install

**Step 3: Install System Requirements**

**Windows Installation:**

* Install **Docker**.
* Install **GraphicsMagick** and **ImageMagick**.
* Install **FFMPEG**.
* Install **Nginx**.
* Install **PM2**.
* Install **Node.js v22**.

**Step 4: Configure the Application**

1. **Create a .env file** in the project root with the following configurations:
2. # Server configuration
3. PORT=3000
4. NODE\_ENV=development
5. # Redis configuration
6. REDIS\_HOST=localhost
7. REDIS\_PORT=6379
8. # File paths
9. UPLOAD\_DIR=uploads
10. PROCESSED\_DIR=processed
11. PUBLIC\_URL=http://localhost:8099
12. # Queue configuration
13. QUEUE\_CONCURRENCY=2
14. **Update Nginx Configuration**
    * Modify the src/config/nginx.conf file with the correct paths (of folders you want to serve) and settings.
    * Replace the existing nginx.conf file in your Nginx installation with this updated file.
15. **Create Required Directories**
16. mkdir -p uploads/images uploads/videos processed/images processed/videos processed/thumbnails

**Step 5: Start the Application**

**Start Nginx:**

cd path/to/nginx

start nginx

**Start Redis (if not running as a service):**

docker-compose up -d

**Start the API Server:**

npm run dev

**Start Workers (Each in a Separate Terminal):**

node src/workers/imageWorker.js

node src/workers/videoWorker.js

**Start with PM2:**

npm run start:pm2

**API Documentation**

**Upload Image**

* **Endpoint:** POST /api/media/image
* **Content-Type:** multipart/form-data
* **Form Parameters:**
  + image: Image file (JPEG, PNG, GIF, WebP)
* **Response:**
* {
* "status": "success",
* "message": "Image processing queued",
* "data": {
* "jobId": "123",
* "fileId": "abc-123",
* "originalName": "profilePic.png",
* "status": "queued"
* }
* }

**Upload Video**

* **Endpoint:** POST /api/media/video
* **Content-Type:** multipart/form-data
* **Form Parameters:**
  + video: Video file (MP4, MOV, AVI, WebM)
* **Response:**
* {
* "status": "success",
* "message": "Video processing queued",
* "data": {
* "jobId": "456",
* "fileId": "def-456",
* "originalName": "example.mp4",
* "status": "queued"
* }
* }

**Get Image Status**

* **Endpoint:** GET /api/media/image/:fileid
* **Response (In Progress):**
* {
* "status": "success",
* "data": {
* "fileId": "abc-123",
* "jobId": "123",
* "status": "active",
* "progress": 50
* }
* }
* **Response (Completed):**
* {
* "status": "success",
* "data": {
* "fileId": "abc-123",
* "status": "completed",
* "url": "http://localhost:8099/processed/images/abc-123.jpg"
* }
* }

**Get Video Status**

* **Endpoint:** GET /api/media/video/:id
* **Response (Completed):**
* {
* "status": "success",
* "data": {
* "fileId": "def-456",
* "status": "completed",
* "videoUrl": "http://localhost:8099/processed/videos/def-456.avi",
* "thumbnailUrl": "http://localhost:8099/processed/thumbnails/def-456.jpg"
* }
* }

**Get File Status**

* **Endpoint:** GET /api/media/status/:fileId
* **Response:**
* {
* "id": "123",
* "fileId": "abc-123",
* "status": "completed",
* "progress": 100
* }

**Retrieve Processed Media**

* **Get All Images:** GET /api/media/images
* **Get All Videos:** GET /api/media/videos

**Conclusion**

The **Fraction Media Processing System** provides a reliable and scalable solution for handling image and video processing. By leveraging Redis for task management, Nginx for efficient file serving, and PM2 for process supervision, this system ensures high performance and reliability.